



Development Tools

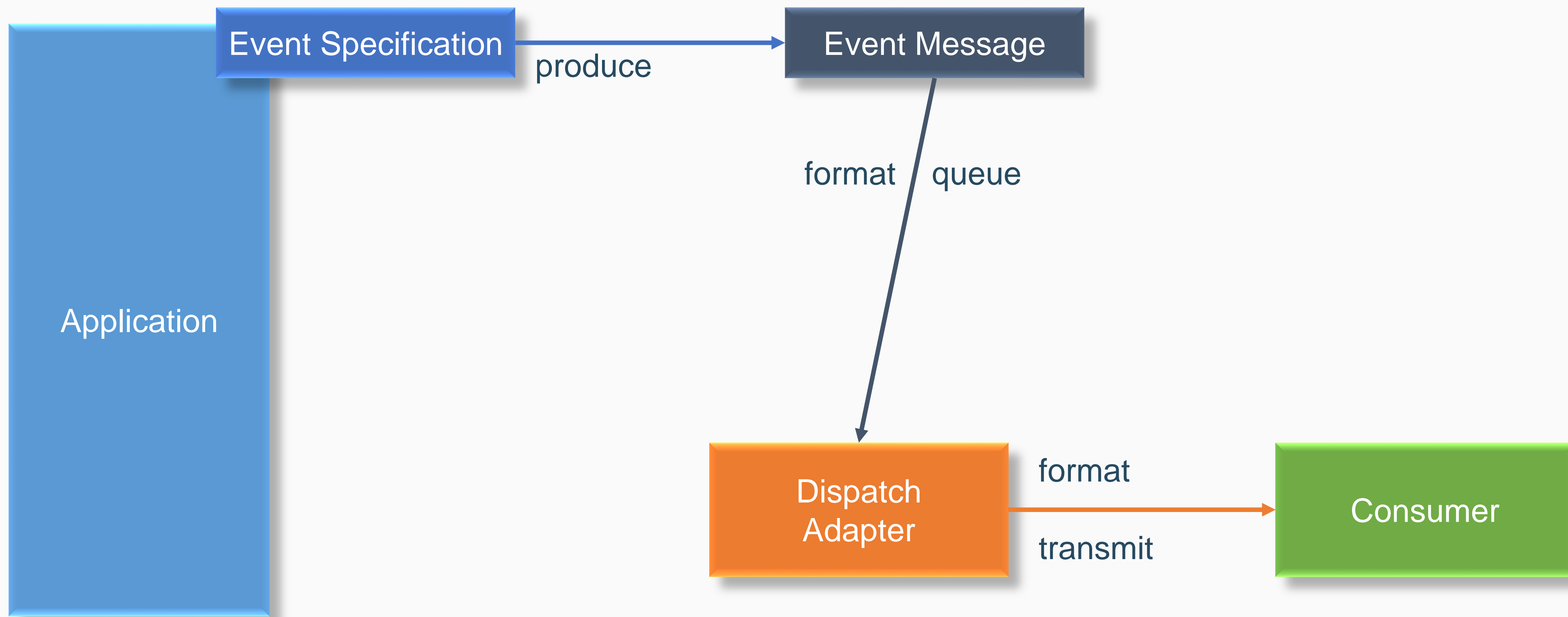
TPF Toolkit and Debugger update

Matt Gritter
TPF Toolkit

© 2017 IBM z/TPF | TPF Users Group Spring Conference | IBM Confidential

IBM z/TPF
April 4th, 2017

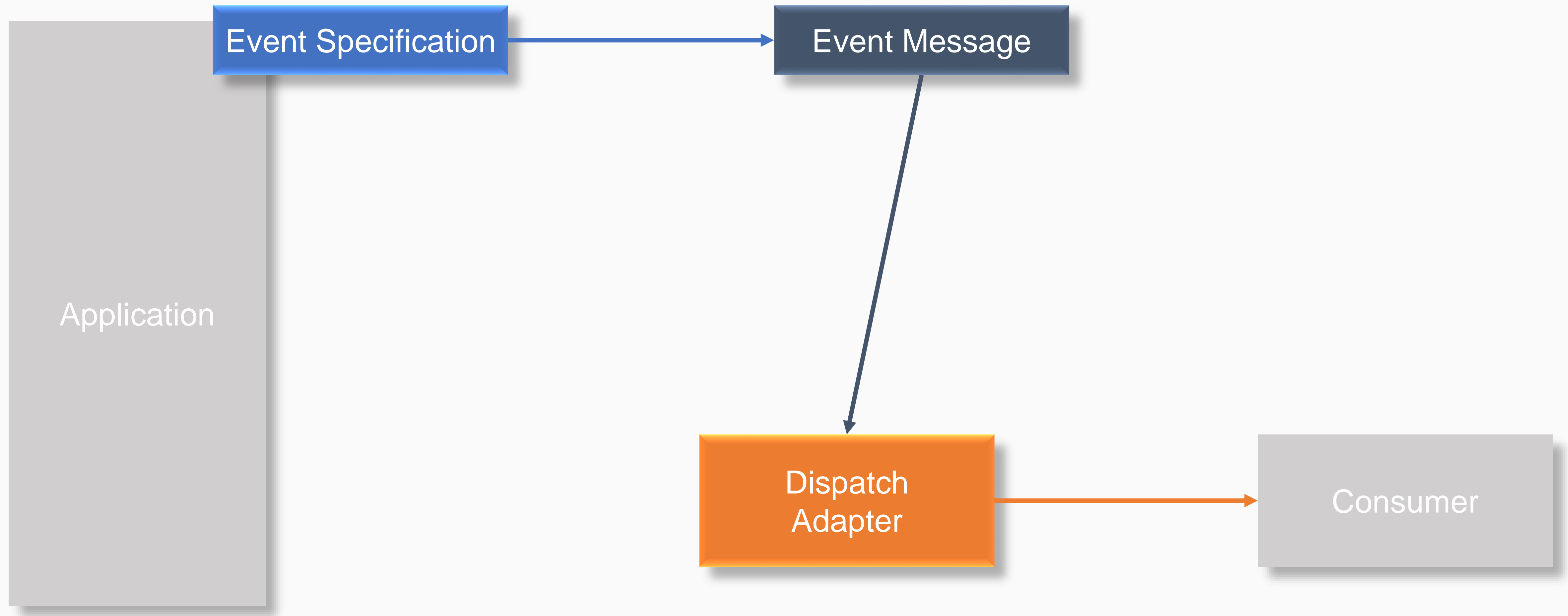
Business Event descriptors overview



Business Event descriptors

Prior to the TPF Toolkit 4.2.7 release, business event deployment descriptors had to be created by hand using template files as reference.

Business Event descriptors



Business Event descriptors

TPF Toolkit 4.2.7 introduced tooling for creating:

- Dispatch Adapters descriptors (XML)
- Data / Signal Event Message descriptors (DFDL)
- Data / Signal Event Specification descriptors (XML)
- z/TPF File Collection descriptors (XML)
- File Event Data descriptors (DFDL)

Business Event descriptors

TPF Toolkit 4.2.7 also included

- DFDL editor
- DFDL parse and serialize testing framework

Business Event descriptors

TPF Toolkit 4.2.9 added

- Validation mechanism to the DFDL editor

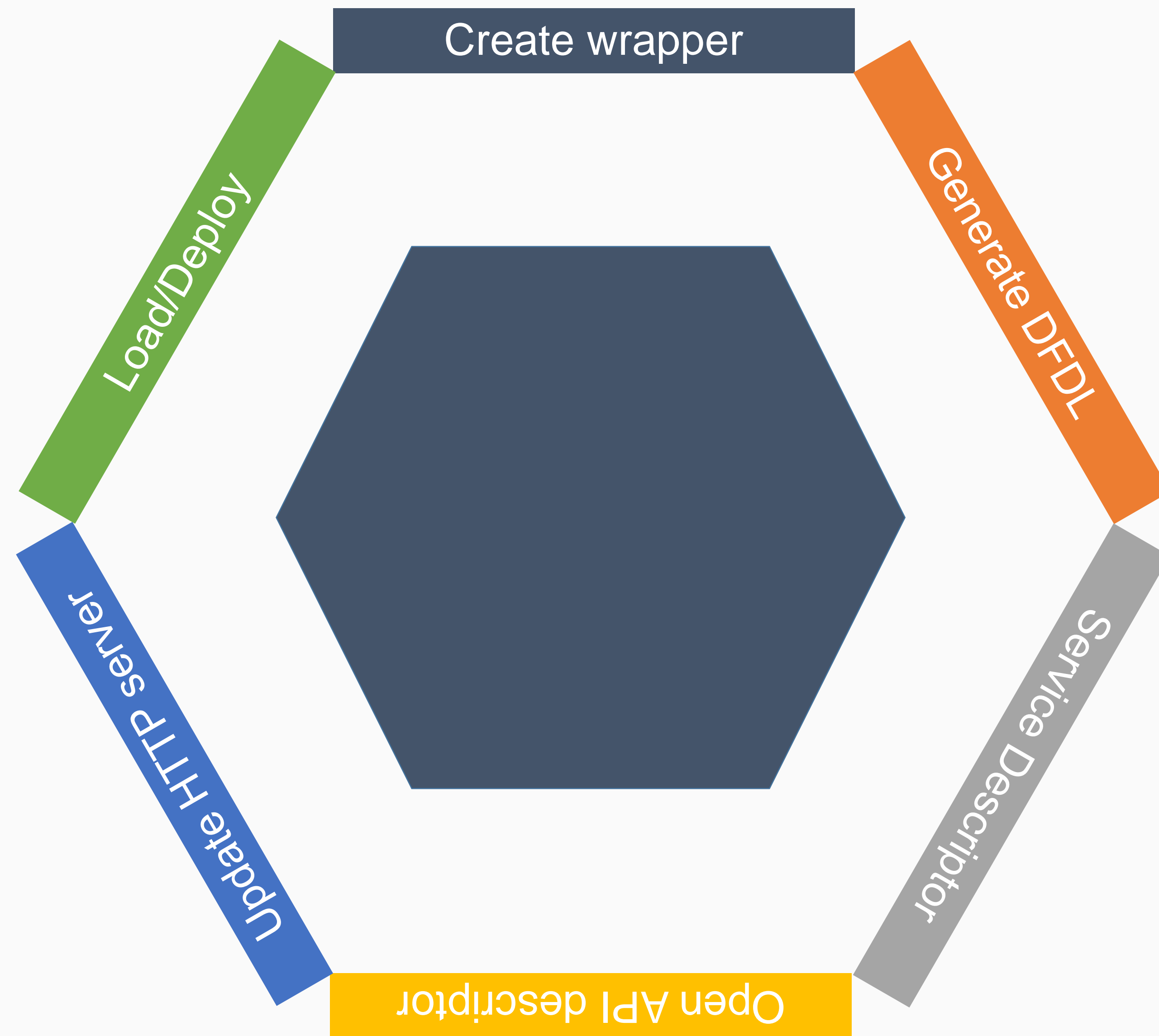
REST service descriptors

Tooling for creating:

- Service Definition descriptors
- Service Archive files (z/OS Connect EE API editor support)

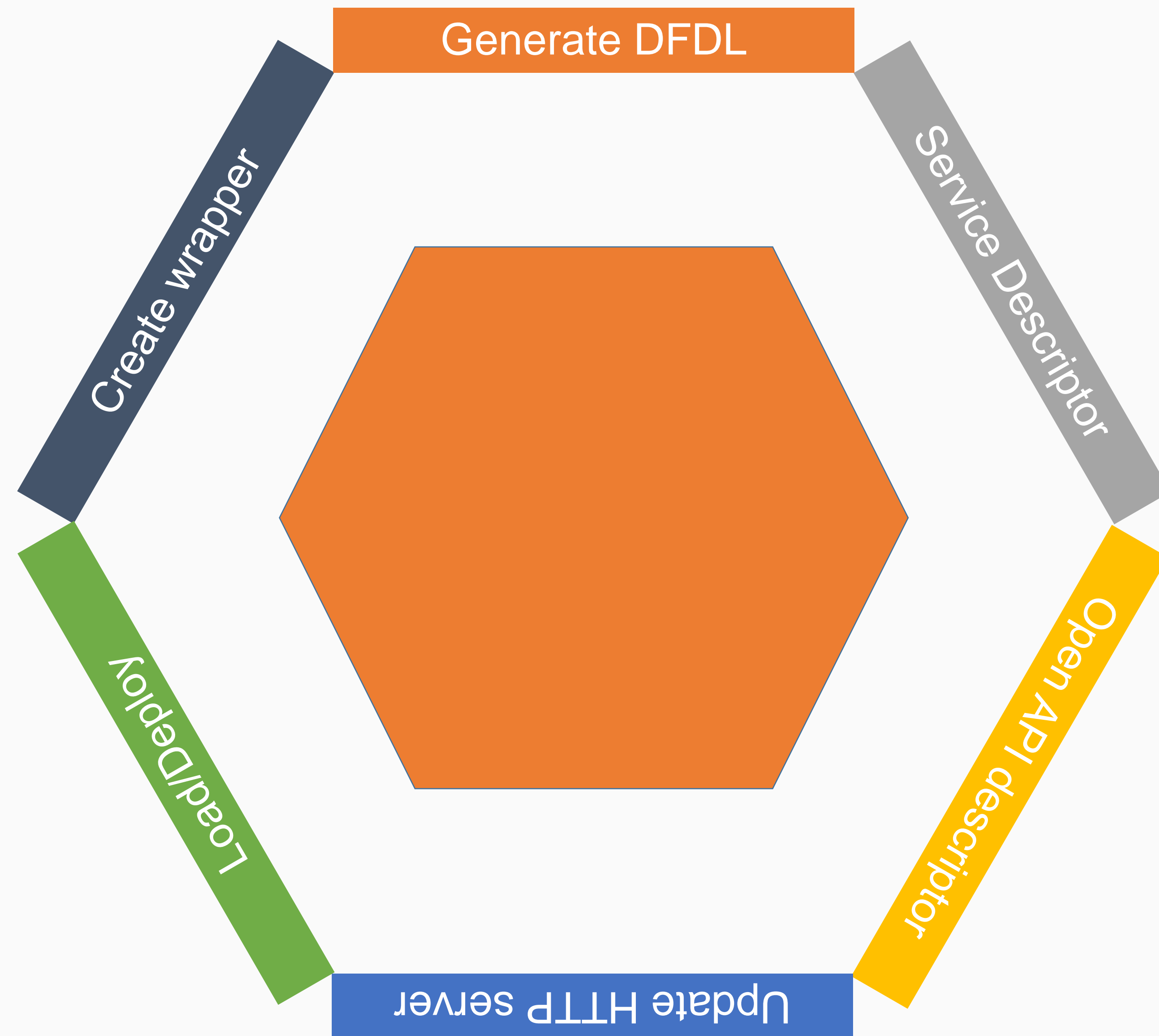
In TPF Toolkit

- Create a wrapper program to enable calling an existing application using a REST service
- Request structure
- Response structure
- Handle request
- Produce response



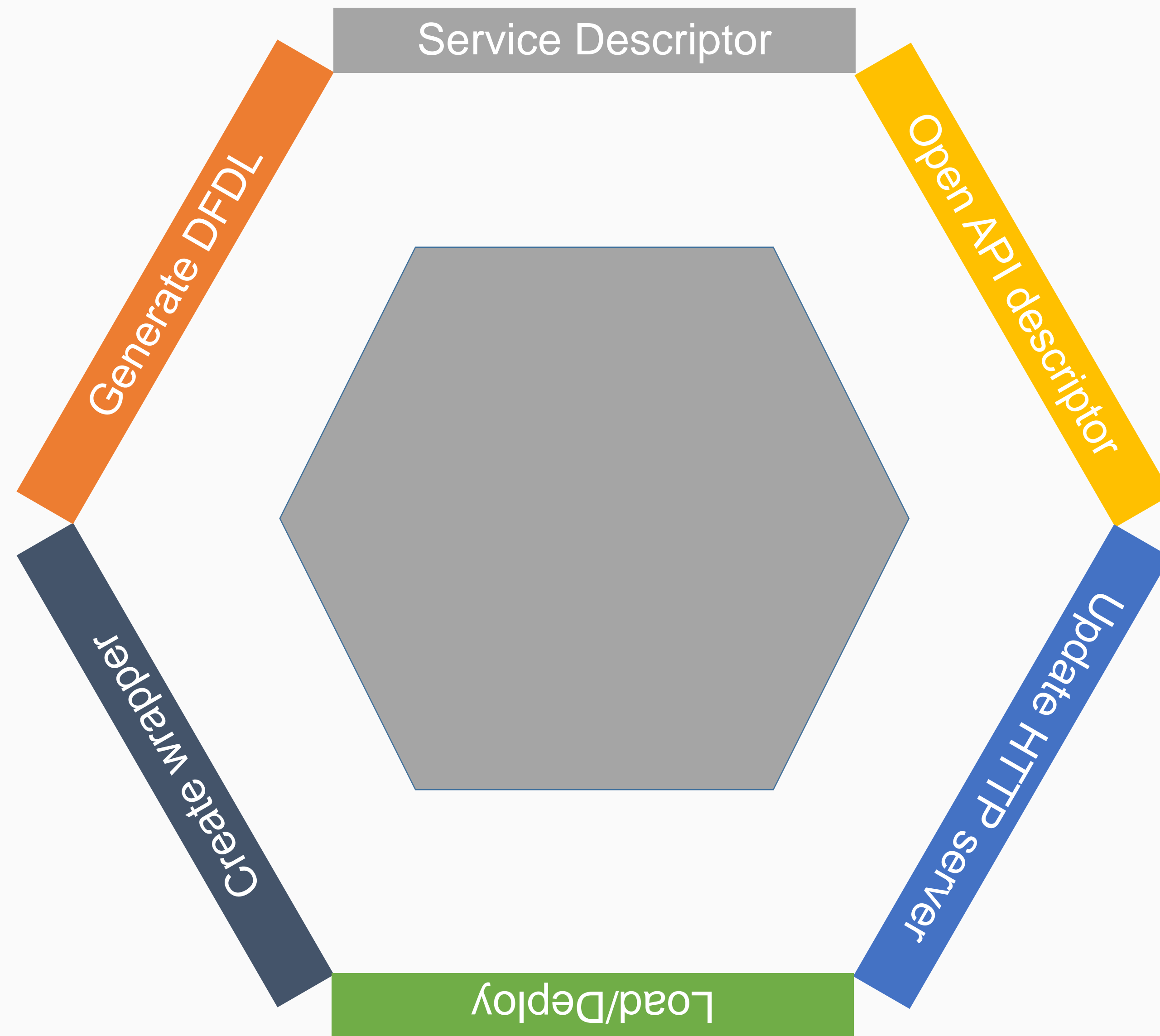
In TPF Toolkit

- Build wrapper code via MakeTPF integration with the Generate DFDL Artifacts menu action
- Import request and response DFDL into the descriptor definition project
- Test parse and serialize generated DFDL using the test framework



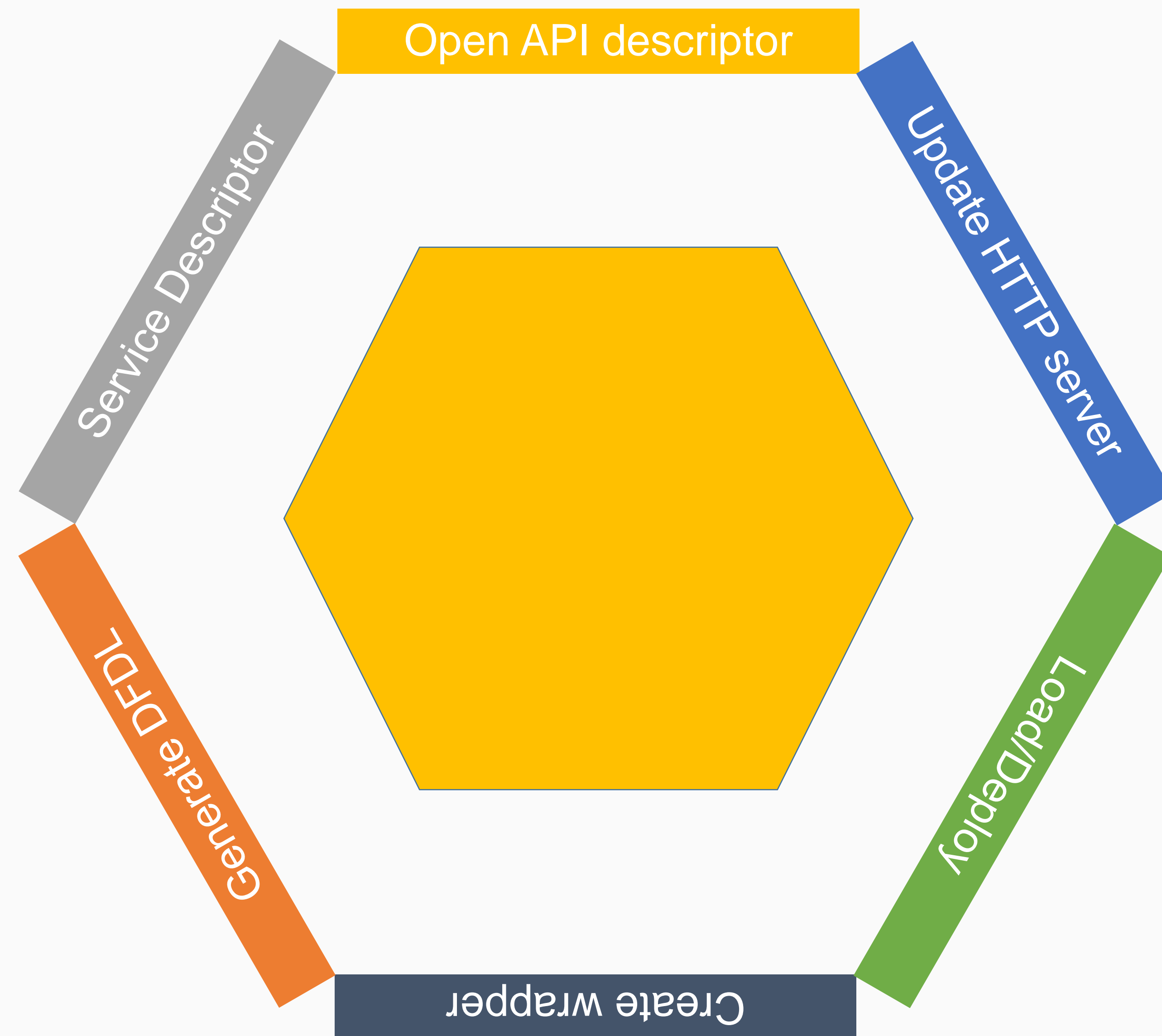
In TPF Toolkit

- Run the New Service Artifacts wizard
- Enter service information
- Supply request and response DFDL as inputs
- DFDL request and response are converted to JSON schema and packaged in a Service Archive file for use with the z/OS Connect EE API Editor



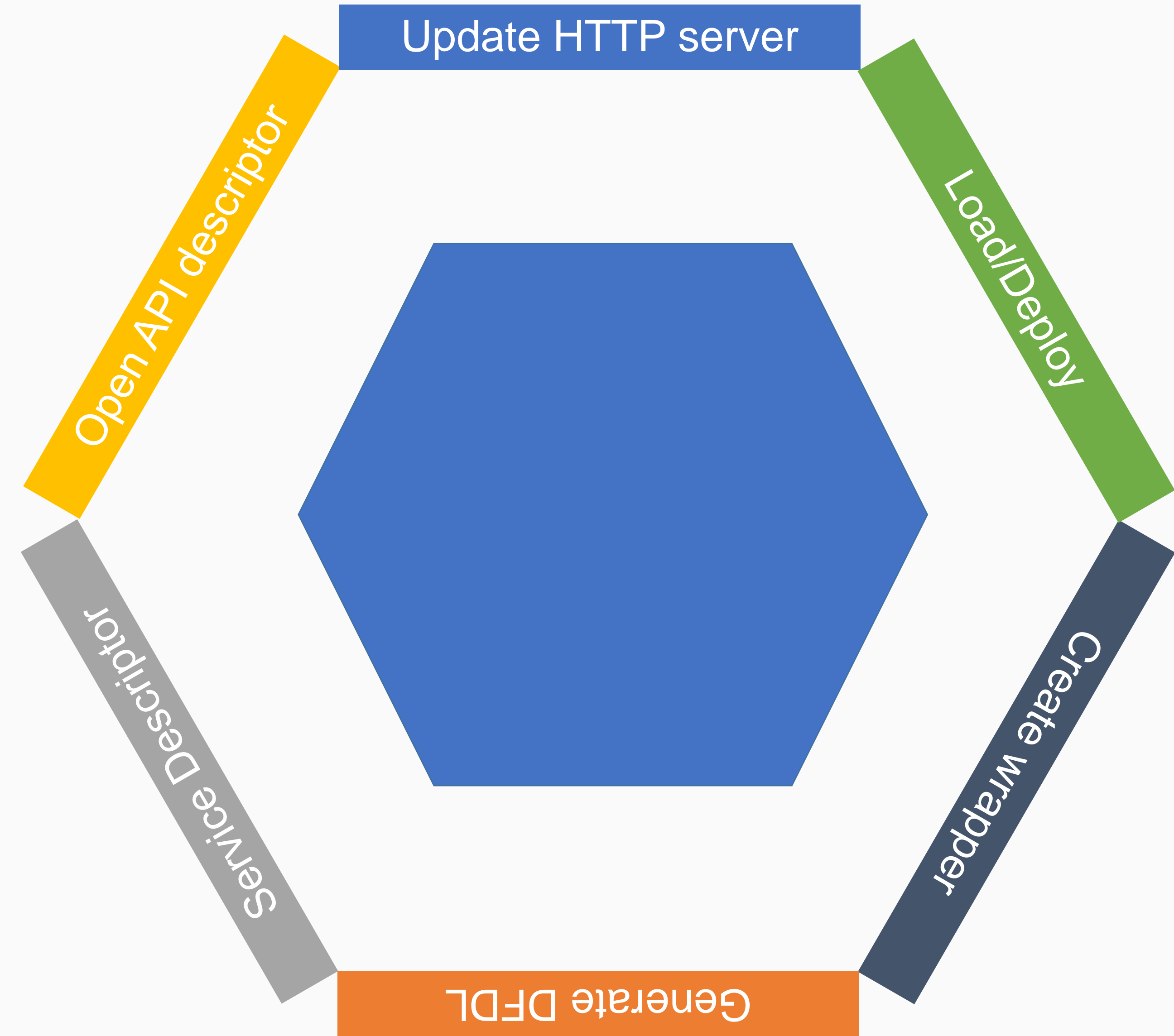
In IBM Explorer for z/OS

- Import the Service Archive file into the z/OS Connect EE API Editor
- Specify paths and methods
- Map parameters to fields in the request
- Export the generated Open API descriptor and import into a descriptor definition project



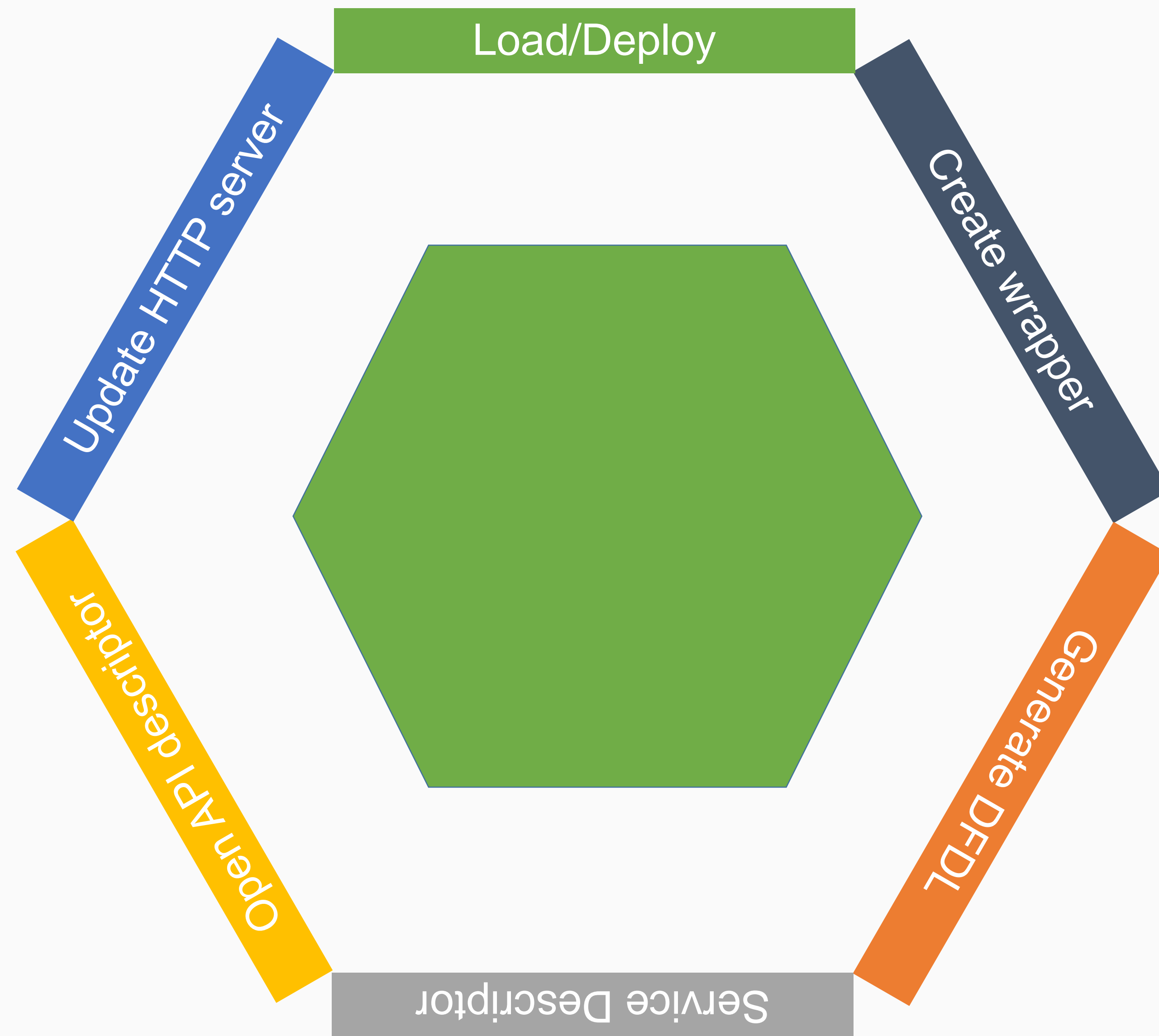
User Action

- Update the URL mapping file for the TPF host system



In TPF Toolkit

- Load wrapper code and service artifacts
- Load using LoadTPF and OLDR actions available from the Descriptor Project Navigator view
- Deploy artifacts as directed in TPF documentation



REST service descriptors

Additional resources

- YouTube: [Creating Native REST Artifacts for z/TPF](#)

Java™ service descriptors

Tooling for creating:

- JAM descriptors for Java based services

Java service descriptors

- Similar workflow for REST services with an additional descriptor to associate the service with a JAM runtime

Java service descriptors

Additional resources

- YouTube: [Creating JAM service artifacts for a native TPF application](#)

SNAPC in dump viewer

- ZASER SNAP commands to enable capture of SNAPC
- Found in the Dump Viewer sub-system of TPF connections in the Remote Systems view
- Debug action available from context-menu
- Display captured state of SNAPC

What's new

The screenshot displays the IBM TPF Toolkit debugger interface. The main window shows a C++ source file named `qdb0er.cpp` with the following code snippet:

```

537
538     snapcEcbList[3]->snapc_len = 0;
539     switch(caseNum)
540     {
541     case 200:
542         snapc(SNAPC_RETURN,0x101111,
543             "snapc issued from C pgm with debug info, case 200 \n",
544             snapc_list_ptr,'A',SNAPC_NOREGS,SNAPC_ECB,NULL);
545
546         QDBDC1F1(type,caseNum);
547
548         snapc(SNAPC_EXIT,0x101222,
549             "snapc issued from C pgm with debug info, case 200 \n",

```

The **Variables** window on the right shows the following data:

Name	Value
caseNum	200
header_buffer	
i	0
j	0
list	
snap_msg	""
snapcEcbList	
snapc_list	
snapc_list_ptr	
type	1

The bottom section of the interface contains the **ECB Tree**, **ECB Dump**, and **ECB EBCDIC** views. The **ECB Dump** view shows a table of memory addresses and their corresponding values:

Address	0 - 3	4 - 7	8 - B	C - F
FC00000	CHW 00000000	BAD 00000000	W00 C801E4C7	004 C3E5E9E9
FC00010	008 80B00000	012 04B51EF8	016 00000000	020 00004E90
FC00020	024 04A05640	028 00000130	032 00000001	036 88F92000
FC00030	040 01000000	044 00001000	048 00000000	052 E2D4D7C1
FC00040	056 010000C1	060 80B00000	064 00000000	068 00005334
FC00050	072 00008400	076 0400FA00	080 E3C5E2E3	084 00002000
FC00060	088 00000000	092 03F578C8	096 00000000	100 00000000

The **ECB EBCDIC** view shows the corresponding EBCDIC characters for the same memory locations:

0 - 3	4 - 7	8 - B	C - F
....	H.UG	CVZZ
0^..	.\$.8+°
.μī	h9.
....	SMPA
...A	0^..ë
..d.	..³.	TEST	...
....	.5İH

What's new

Description	z/TPF APAR	TPF Toolkit version	Requirement
Dump viewer for SNAPC	PJ43583	4.2.7	RFE 64974
Data Event descriptor projects	PJ44028	4.2.7	Customer Request
REST descriptor projects	PJ44281	4.2.9	Customer Request

YouTube channel

How to find it:

- [Link](#) in this presentation
- Search for IBM TPF Toolkit on YouTube
- TPF Toolkit documentation
- TPF Blog posts

Any reference to future plans are for planning purposes only.

IBM reserves the right to change those plans at its discretion.

Any reliance on such a disclosure is solely at your own risk.

IBM makes no commitment to provide additional information in the future.

Scriptable Code Coverage

Goal:

- allow autonomous collection of code coverage for TPF applications

Value:

- Integration point for automated testing
- Integration point with Application Delivery Intelligence for trend analysis
- See TPF Toolkit task force presentation for full details

Next Major Release of TPF Toolkit

- Eclipse 4.6 (Neon)
- Java 8
- Simplified installation mechanism
- Synchronized projects replacing RSE DataStore server
- Improved Eclipse C/C++ Development Tools support
- Focus on usability

Simplified installation mechanism

- Unzip and go
- Eclipse P2 update mechanism
- Simpler third party Eclipse tools integration

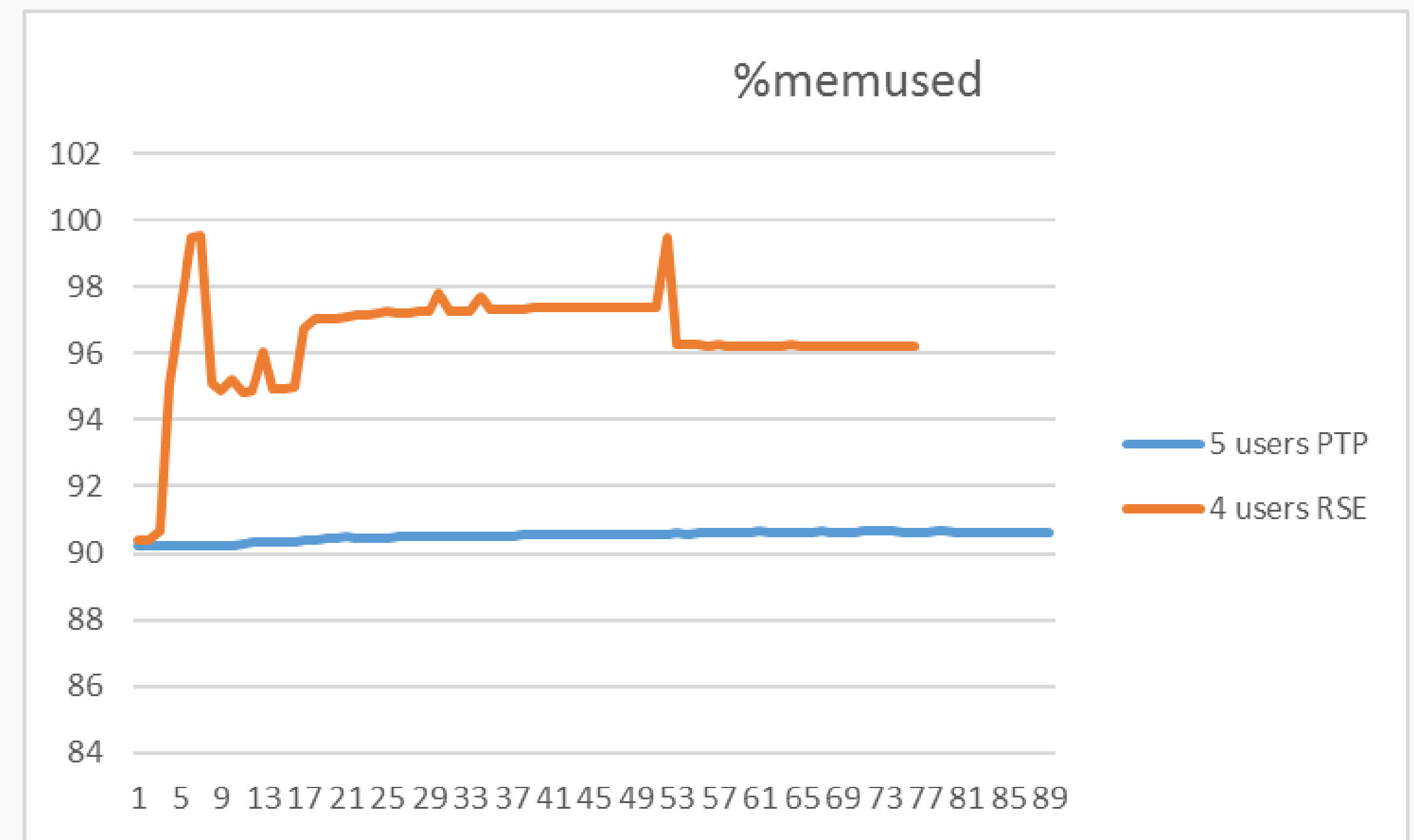
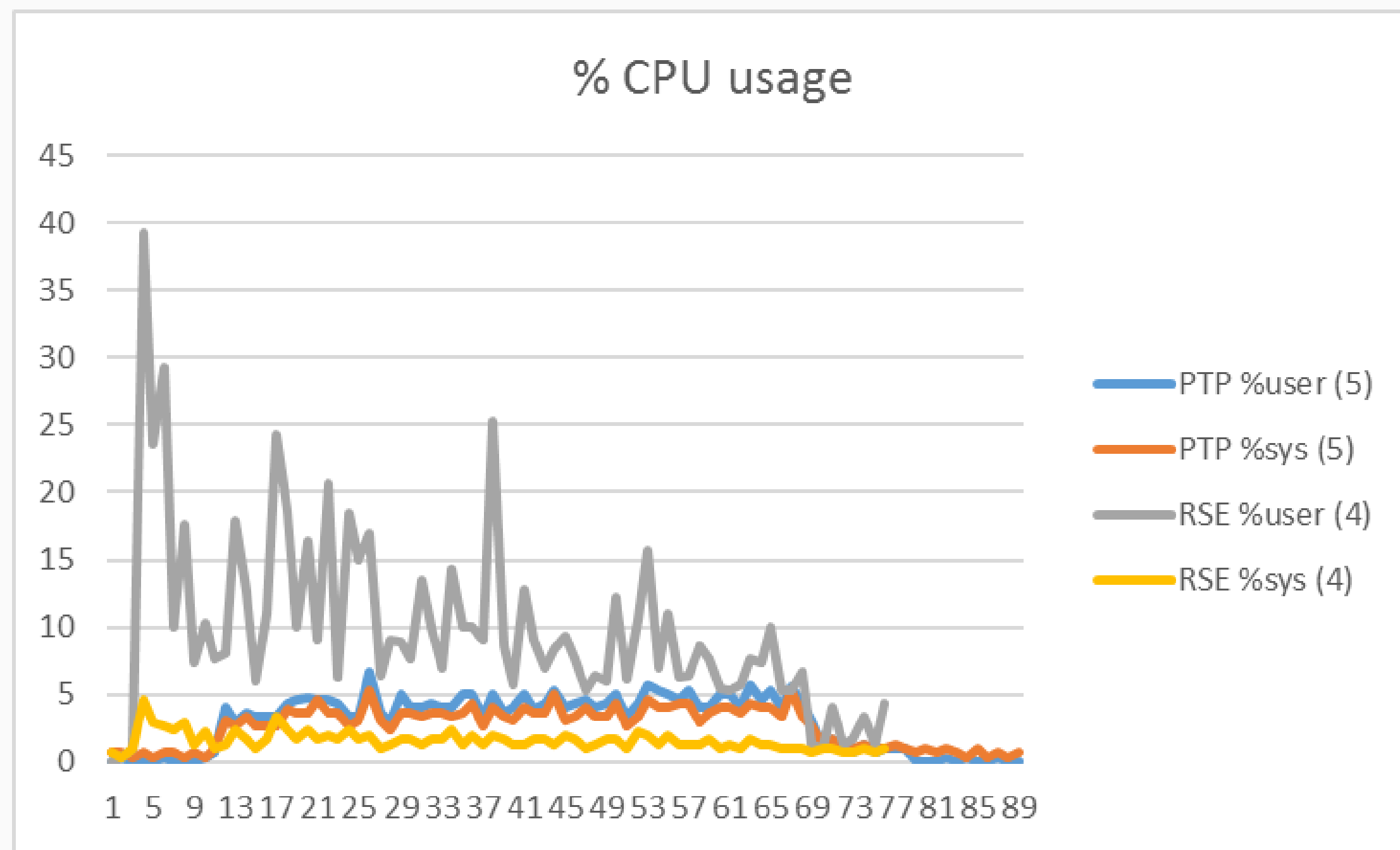
Synchronized projects

- Project code is kept locally and synchronized automatically or on demand to the remote system.
- Custom filtering when synchronizing to ignore any unwanted files
- Possible to setup synchronization points to multiple systems
- Changes from remote system synchronized
- Improved conflict resolution if local and remote both have changes

Synchronized projects – performance test

Small test system with 4 to 5 concurrent users

- running the same automated scenario
- updating and saving the same file every 5 to 10 seconds



Improved Eclipse C/C++ Development Tools support

- Remote C/C++ indexer enables accessing remote header files that are not contained within the project space
- Jump to remote headers from project source
- Auto-completion support based on the remote index content
- Improved Outline view experience
- And more...

Focus on usability

You have the opportunity to participate in the development of the next major version of TPF Toolkit!

If you are interested participating in design thinking sessions with the TPF Toolkit team or receiving beta releases, let us know!



THANK YOU

Questions or comments?

Matt Gritter
TPF Toolkit

© 2017 IBM z/TPF | TPF Users Group Spring Conference | IBM Confidential

IBM z/TPF
April 4th, 2017

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Notes

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.